6. *(Amended)* Arrangement as claimed in [any of the preceding claims] claim 1, characterized in that an engineering model is developed by letting each computational object (CO) be mapped to one or more Basic Engineering Objects (BEOs), interaction between computational objects (CO1, CO2) belonging to the same cluster being effected directly as method calls, and communications between [computunal] computational objects (CO3, CO4) located in the telecom system domain and in different clusters being effected through a channel comprising stubs, binders and protocols.

7. *(Amended)* Arrangement as claimed in [any of the preceding claims] claim 1, characterized in that communication between user domain computation objects (CO2) and telecom system domain computation object (CO3) residing in different clusters will take place in a channel comprising an interceptor ( ) in addition to the usual objects, said interceptor being invisible for the application designer.

8. *(Amended)* Arrangement as claimed in [any of the preceding claims] claim 1, characterized in that from the computational model the application designer can decide whether an object belongs to the user domain or the telecom system domain, for on the basis of such a decision to generate the necessary objects, such as stubs, for all the application objects.

9. *(Amended)* Arrangement as claimed in [any of claims 1-2] claim 1, characterized in that the arrangement comprises means for sending a message to a routing broker asking for specific server to perform a task, said broker locating the server and sending said request to said server, said mobility functions (M) thereby playing the role as the broker.

10. *(Amended)* Arrangement as claimed in [any of the claims 1-2 or 9] claim 1, characterized in that said mobility function (M) comprises a cascade of two brokers.

11. *(Amended)* Arrangement as claimed in [any of the claims 1-2 or 9-10] claim 1, characterized in that said mobility functions (M) in the form of two brokers allows for interaction between an object (CO1) belonging to a user domain and an object (CO2)

belonging to a telecom system domain by letting said interactions enter said mobility functions (M) at one end and exit therefrom at the other end [(Fig. 5)].

12. *(Amended)* Arrangement as claimed in [any of the claims 1-2 or 9-11] claim 1, characterized in that both ends of the mobility functions (M) support both the same interface type containing an "invoke type" operation, for thereby allowing a request to be built and invoked dynamically by client objects.

14. *(Amended)* Arrangement as claimed in [any of the claims 1-2, or 9-13] claim 1, characterized in that the mobility functions (M) are introduced in the system architecture as a functional layer between the application layer and the DPE layer, said layer separation allowing every layers to use services offered by other layers.

15. *(Amended)* Arrangement as claimed in [any of the claims 1-2, or 9-14] claim 1, characterized in that there is introduced an intermediary model called derived computational model in the transition from the computational model to the engineering model, said intermediary model being used to map all interactions traversing the boundary between the user domain and the telecom system domain to interactions with the mobility functions (M).

16. *(Amended)* Arrangement as claimed in claim 15, characterized in that on the basis of the derived computational model, an engineering model can be elaborated and engineering objects, such as stubs, can be mechanically generated.

17. *(Amended)* Arrangement as claimed in [any of the claims 1-2] claim 1, characterized in that said arrangement comprises proxy objects, a proxy acting on behalf of an entity in a transparent way, i.e., when interacting with a proxy an entity cannot tell whether it is dealing with the real counterpart or with its proxy.

21. *(Amended)* Arrangement as claimed in [any of the claims 1-2 or 17-20], characterized in that there is introduced an object type designated Dynamic Object (DO), all proxies being of the type DO, and that a proxy, i.e., an object (instance) of type DO is initiated from an object template called Dynamic Object Implementation (DOI).